EXHIBIT G

5.0 Reference

This section describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the XMC Driver Administrator Component.

5.1 Interfaces

Other than the standard OLE interfaces **IUnknown**, **IClassFactory**, and **IDispatch**, there is one other standard OLE interface, and two custom OLE interfaces specific to the XMC Driver Administrator Component. These interfaces are the **IXMC_EnumDriver**, **IXMC_DriverAdmin**, and **IXMC_DriverAdminDebug** interfaces.

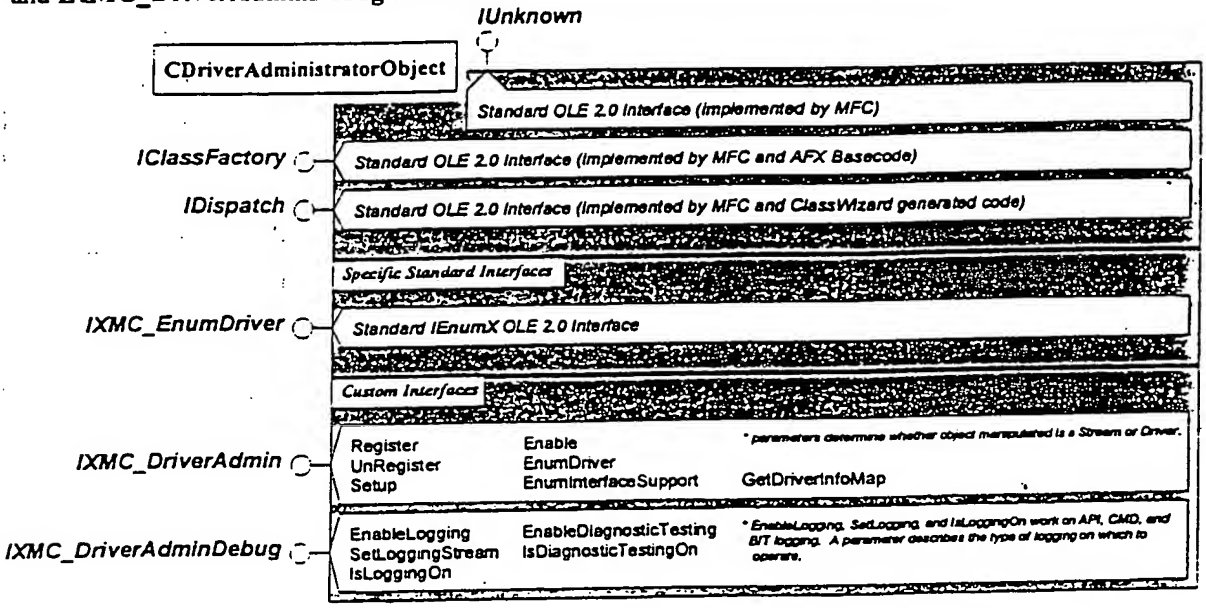

Figure 11 *Interface-Map.*

For more information on the standard OLE interfaces see the *OLE 2 Programmer's Reference*. And, for more information on the custom XMCAPI interfaces, see the *XMCAPI Reference*.

5.2 Exported Functions

The following are the functions exported by the driver DLL.

```
XMC_DRIVERADMIN_MODULETYPE      DLLGetModuleType( void );
LPCLSID                         DLLGetCLSID( void );
BOOL                            DLLRegisterServer( void );
BOOL                            DLLUnRegisterServer( void );
```

5.3 Structures and Defines

This section defines all structures, enumerations, and defines used by the driver.

5.3.1 XMC_DRIVERADMIN_MODULETYPE Enumeration

The following enumeration is used to mark the module type of the component.

```
enum XMC_DRIVERADMIN_MODULETYPE
{
        XMC_DRIVERADMIN_MT   = 0x6000
};
```

5.3.2 XMC_LOGGINGTYPE Enumeration

The following enumeration describes the different types of logging supported by the XMC Driver Administrator Component.

```
enum XMC_LOGGINGTYPE
{
        XMC_LT_API,
        XMC_LT_CMD,
        XMC_LT_BIT
};
```

5.3.3 XMC_DRIVER_INFO Structure

The following structure is used when setting up and querying the state of the driver.

```
struct XMC_DRIVER_INFO
{
        XMC_HDRIVER                 [in,out]    m_hDriver;
        XMC_DRIVER_MODULETYPE       [out]       m_mt;
        BOOL                        [out]       m_bStreamMgmtLocked;
        LPTSTR                      [out]       m_pszHardwareVendor;
        DWORD                       [in]        m_cbMaxHardwareVendor;
        LPTSTR                      [out]       m_pszHardwareModel;
        DWORD                       [in]        m_cbMaxHardwareModel;
        LPTSTR                      [out]       m_pszDriverVendor;
        DWORD                       [in]        m_cbMaxDriverVendor;
};
```

5.3.4 XMC_STREAM_INFO Structure

This structure is used to pass all stream specific data used to both setup the stream, and query the stream for its current settings.

```
struct XMC_STREAM_INFO
{
        XMC_HSTREAM                 m_hStream;
        XMC_STREAM_MODULETYPE       m_mt;

        union {
                struct PCBus
                {
                        DWORD dwPort;
                        DWORD dwIRQ;
                },

                struct Serial
                {
                        DWORD dwPort;
                        DWORD dwBPS;
                },

                struct TextFile
                {
                        LPCTSTR pszFileName;
                },

                struct Custom
                {
                        DWORD dwParam1;
                        DWORD dwParam2;
                },
        };
};
```

5.4 Class s

This section contains the definition of all classes used by the XMC Driver Administrator Component in its implementation.

5.4.1 CDriverAdminDisp Class

The **CDriverAdminDisp** class acts as the dispatch, or control, object used to separate all OLE details from the code implementing the driver internals. This object coordinates all operations taking place in the driver by directing other C++ object to carry out the work of the operation. The following is the definition of the **CDriverAdminDisp** class.

```
class CDriverAdminDisp
{
      public:
            //---- Constructors & Destructors ----

            CDriverAdminDisp( void );
            ~CDriverAdminDisp( void );


            //---- Initialization ----

            DWORD Initialize( void );


            //---- IXMC_EnumDriver Helper Methods ----

            DWORD GetFirstDriverCLSID( LPCLSID pDrvCLSID );
            DWROD GetNextDriverCLSID( LPCLSID pDrvCLSID );
            DWORD GetFirstDriver( LPUNKNOWN *ppDriverUnk );
            DWORD GetNextDriver( LPUNKOWN *ppDriverUnk );


            //---- IXMC_EnumInterfaceSupport Helper Methods ----

            DWORD GetFirstInterfaceSupport( LPDRVIFSUPPORT pDIFS );
            DWORD GetNextDriverInterfaceSupport( LPDRVIFSUPPORT pDIFS );


            //---- IXMC_DriverAdmin Methods ----

            DWORD GetDriverInfoMap( LPXMCDRIVERINFOMAP pDIMap );
            DWORD Register( XMC_HDRIVER hDrv, XMC_HSTREAM hStrm,
                            LPCTSTR pszModuleName );
            DWORD UnRegister( XMC_HDRIVER hDrv, XMC_HSTREAM hStrm );
            DWORD Setup( XMC_HDRIVER hDrv, XMC_HSTREAM hStrm,
                            LPVOID pInfo );
            DWORD Enable( XMC_HDRIVER hDrv, XMC_HSTREAM hStrm );


            //---- IXMC_DriverAdminDebug Methods ----

            DWORD EnableLogging( XMC_LOGGINGTYPE lt, BOOL bEnable );
            DWORD SetLoggingStream( XMC_LOGGINGTYPE lt,
                            LPXMCSTREAM pStrm );
            BOOL  IsLoggingOn( XMC_LOGGINGTYPE lt );
            DWORD EnableDiagnosticTesting( BOOL bEnable );
            BOOL  IsDiagnosticTestingOn( void );


      private:
            //---- Privat  Data ----

            CModuleMgr m_moduleMgr;
};
```

5.4.2 CModuleMgr Class

The CModuleMgr is responsible for managing all drivers and streams registered in the system. For example, stream and driver registration is controlled by the CModuleMgr class. A list of all drivers registered with the CModuleMgr is stored in the registration database. Stream registration is passed through to the corresponding driver. The CModuleMgr class is defined as follows.

```
class CModuleMgr
{
      public:
            //---- Constructors & Destructors ----

            CModuleMgr( void );
            ~CModuleMgr( void );


            //---- Initialization ----

            DWORD Initialize( void );


            //---- Driver Actions ----

            DWORD RegisterDriver( XMC_HDRIVER hDriver );
            DWORD UnRegisterDriver( XMC_HDRIVER hDriver );
            DWORD SetDriverInfo( XMC_HDRIVER hDrv, LPXMCDRIVERINFO pDI
);
            DWORD GetDriverInfo( XMC_HDRIVER hDrv, LPXMCDRIVERINFO pDI
);


            //---- Stream Actions ----

            DWROD RegisterStream( XMC_HSTREAM hStream );
            DWORD UnRegisterStream( XMC_HSTREAM hStream );
            DWORD SetStreamInfo( XMC_HSTREAM hStrm, LPXMCSTREAMINFO pSI );
            DWORD GetStreamInfo( XMC_HSTREAM hStrm, LPXMCSTREAMINFO pSI );


            //---- General Actions ----

            DWORD GetFirstInterfaceSupport( LPDRVIFSUPPORT pDIFS );
            DWORD GetNextDriverInterfaceSupport( LPDRVIFSUPPORT pDIFS );
            DWORD GetFirstDriverCLSID( LPCLSID pDriverCLSID );
            DWORD GetNextDriverCLSID( LPCLSID pDriverCLSID );
            DWORD GetFirstDriver( LPUNKNOWN *ppDriverUnk );
            DWORD GetNextDriver( LPUNKNOWN *ppDriverUnk );
            DWORD LoadDriverInfoMap( LPDRIVERINFOMAP pDrvInfoMap );

      private:
            //---- Private Data ----

            CSimpleDriver       m_rgDrivers[];
            DWORD               m_cbDrivers;
};
```

5.4.3 CSimpleDriver Class

The CSimpleDriver class is used to communicate directly with the driver component. All OLE related details are encapsulated within this class.

```
class CSimpleDriver
{
      public:
            //---- Constructors & Destructors ----

            CSimpleDriver( void );
```

```
            ~CSimpleDriver( void );


            //---- Initialization ----

            DWORD Initialize( CLSID& rDriverCLSID );
            DWORD Attach( CLSID& rDriverCLSID );
            DWORD Detach( void );


            //---- Actions ----

            DWORD SetDriverInfo( LPXMCDRIVERINFO pDI );
            DWORD GetDriverInfo( LPXMCDRIVERINFO pDI );
            DWORD GetStreamList( LPSIMPLESTREAM rgStrms, LPDWORD pdwCount );
            DWORD GetCLSID( LPCLSID pDriverCLSID );
            DWORD GetHandle( LPXMC_HDRIVER phDriver );

      private:
            //---- Private Data ----

            LPXMCDRVCORESTATICSTATE         m_pDrvCoreStaticState;
            LPXMCDRVCOREDYNAMICSTATE        m_pDrvCoreDynamicState;
            LPXMCDRVEXTSTREAMMGMT           m_pDrvExtStreamMgmt;
};
```

5.4.4 CSimpleStream Class

The CSimpleStream class is used to communicate directly with the stream component. All OLE related details are encapsulated within this class.

```
class CSimpleStream
{
      public:
            //---- Constructors & Destructors ----

            CSimpleStream( void );
            ~CSimpleStream( void );


            //---- Initialization ----

            DWORD Initialize( CLSID& rStreamCLSID );
            DWORD Attach( CLSID& rStreamCLSID );
            DWORD Detach( void );


            //---- Actions ----

            DWORD SetStreamInfo( LPXMCSTREAMINFO pSI );
            DWORD GetStreamInfo( LPXMCSTREAMINFO pSI );
            DWORD GetCLSID( LPCLSID pDriverCLSID );
            DWORD GetHandle( LPXMC_HSTREAM phStream );

      private:
            //---- Private Data ----

            LPXMCSTREAMINIT           m_pStreamInit;
            LPXMCSTREAM               m_pStream;
};
```

5.4.5 CDriverInfoMap Class

The **CDriverInfoMap** class describes the current state of the driver administrator component regarding installed drivers and streams. The driver administrator CPL queries the driver administrator component for this block of information, that it then uses to update the user-interface describing the drivers and streams installed.

```
class CDriverInfoMap
{
      public:
            //---- Constructors & Destructors ----

            CDriverInfoMap( void );
            ~CDriverInfoMap( void );


            //---- Actions ----

            DWORD Load( LPUNKNOWN *rgpDrvUnk, DWORD dwDrvCount,
                        LPUNKNOWN *rgpStrmUnk, DWORD dwStrmCount );


            //---- Public Data ----

            LPXMCDRIVERINFO     m_rgDriverInfo[];
            DWORD               m_cbDriverInfo;
};
```

5.5 Registration Database Data

The driver administrator manages all drivers registered in the XMC system and stores a persistent list of information, describing each, in the registration database. Below, is the definition of the format used to store the driver information in the registry.

```
XMC.DriverAdministrator.100
    |---- Drivers
            |---- Enabled
            |         |---- nCount  = 1
            |         |---- {CLSID} = "AT6400"
            |
            |---- Disabled
                      |---- nCount  = 2
                      |---- {CLSID} = "DMC1000"
                      |---- {CLSID} = "DT2000"
```